INSTRUCTIONS :

1. Complete all questions in your designated project group.
2. All members must contribute to writing the code. (i.e., 1 question = 1 person, and share the workload if there's an additional question relative to the actual number of members in your team (i.e., 5)). Ensure that all members must understand and explain the codes in any of the questions.
3. During the viva, all students in each team will be randomly asked to describe, answer, and edit any of the answers provided. Marks will be given for your ability to present the answers.

Lab Report

Prepare a report to solve the above problems. The report should contain all the sections as below for each question:

Section	Description
1. Problem	Description of the problem
2. Solution	Explanation on how to solve the problems below
3. Sample Input & Output	A few sets of input and output (snapshot)
4. Source code	Java source code

Requirements

1. Group Assignment (Grouping is the same as your project group)
2. Cover page that includes all student matric numbers and full name.
3. Font: Times New Roman 12, Line Spacing: 1 ½ Spacing
4. Submit to Spectrum according to your OCC. **Deadline**: Before your viva session (W6).

Q1. My Study Planner

Stepping into university life can be quite challenging, but handling these challenges well can turn the same challenges into opportunities. If you are a new student, you will need to balance your academics, sleep, and reflections along with your social life! It's always a good idea to plan your day in advance so that you can stay organized.

In this lab, you have to create a **simple Study Planner Program** that can give you an idea about how productive your day was. Everyone needs a bit of motivation to carry on, don't they? The program should give a motivational message **based on how long you have studied and all the activities you have completed**.

The question will utilize input validation and **creative flow control using nested if-else and optional switch statements** for the evaluation of students' productivity and suggest **improvements based on the productivity score**.

Task

1. Write a program that prompts the user to input:
    - Hours studied
    - Hours slept
    - Number of breaks taken
2. Create a productivity score according to a suitable formula, such as ***score = (hours_studied * 10) - (breaks * 2),*** or you can create your own formula as well, and are open to adding more factors.
3. Display a motivational message based on the score you get. You are open to creativity here. Using the above formula, some of the motivational messages can be:
    - **score > 60** → "Excellent! Keep up the great work!"
    - **30 ≤ score ≤ 60** → "Good effort, but balance your breaks!"
    - **score < 30** → "Try to plan your day better!"
4. Validate the user inputs such that no negative numbers are added. You can define a range here as well.
5. Add a random "Tip of the Day" message using a simple switch structure.

Sample Input

```
Enter hours studied: 6
Enter hours slept: 8
Enter breaks taken: 2
```

Sample Output

```
Your productivity score is 56
Good effort, but balance your breaks!
Tip of the Day: Revise your notes before sleeping.
```

Constraints

- $0 \leq$ hours_studied ≤ 24
- $0 \leq$ hours_slept ≤ 24
- $0 \leq$ breaks ≤ 10
- All inputs must be integers or floats.
- Negative or unrealistic values should display an error message.

Q2. Water Usage Tracker

Malaysia is encouraging households to reduce water wastage. You are tasked to help the city council develop a water billing system that will be able to **calculate the monthly water bills for residential and commercial customers,** considering the amount of water used.

The problem utilizes **conditional statements, mathematical operations, and looping** to demonstrate how continuous billing works for several users.

Task

1. Ask the user for:
    - Customer type (Check if they fall in the category of residential or commercial)
    - Get the data for the liters of water consumed by the user

2. After acquiring the data, apply the tiered billing rules that can vary depending on the specified category.

For the residential category,

- Up to 1000 L → RM 0.02 per L
- 1001–5000 L → RM 0.05 per L
- Above 5000 L → RM 0.08 per L

For the commercial category,

- Up to 10000 L → RM 0.06 per L
- Above 10000 L → RM 0.10 per L

3. Calculate the total bill using the proper conditioning logic.
4. Enabling repeated entry for multiple customers until the user enters “0” to quit.
5. Display the total bill clearly after each entry

Sample Input

```
Customer Type: residential
Liters Used: 3000
```

Sample Output

```
Customer Type: Residential
Water Usage: 3000 L
Total Bill: RM150.00
Thank you for conserving water!
```

Constraints

- 0 ≤ liters ≤ 50000
- customerType must match either “residential” or “commercial” (case-insensitive).
- Invalid or negative inputs must be rejected with an error message.
- Use a while loop to continue billing until the user enters “0” liters.

- Use equalsIgnoreCase() (in Java) or equivalent string comparison for validation.

Q3. Avengers: Secret Binary Code Mission

The Avengers just intercepted a secret encoded message in binary (0s and 1s)! What is your mission? You need to decode the binary sequence and afterward classify which superhero team it belongs to. Lastly, calculate the team's average power.

The question utilizes the knowledge of **logic, loops, and arithmetic** in a creative superhero scenario, demonstrating how flow control can drive decision-making.

Task

1. Prompt the user to enter multiple binary strings. Some of the strings that can be used are 1011, 1100, 1110, or more of your choice.
2. Convert each binary string into its decimal equivalent using loops and arithmetic.
3. Classify the code as:
    - Even → Iron Man's Team
    - Odd → Captain America's Team
4. Keep the count of codes for each team. Use that count to calculate the average power, which will be in decimal value.
5. When the user types "STOP", stop taking the input.
6. Display final stats and team analysis.

Sample Input

```
Enter binary code (type 'STOP' to finish):
1011
1100
1110
STOP
```

Sample Output

```
Code 1011 → Decimal 11 → Captain America’s Team
Code 1100 → Decimal 12 → Iron Man’s Team
Code 1110 → Decimal 14 → Iron Man’s Team

Iron Man Team: 2 codes | Average Power: 13
Captain America Team: 1 code | Average Power: 11
Mission Accomplished: Message Decrypted!
```

Constraints

- Each binary string must only contain 0 and 1.
- Per run, there can only be a maximum of 20 binary codes.
- Input ends when "STOP" is entered (case-insensitive).
- Invalid binary codes must trigger an error message (e.g., “Invalid code: must contain only 0s and 1s”).
- Use loops (while or for) and arithmetic to convert binary manually (no built-in converters).

Q4. Factorial Applications in Computer Science

Factorials are an essential mathematical concept in computer science, used in combinatorics, data analysis, cryptography, and algorithm design. They help determine how many ways items can be arranged or combined, which is vital for solving problems involving probability, security systems, and game logic.

In this lab, you will write a program that calculates the factorial of a number using loops and apply it to real-world computing scenarios such as combinatorics, security, or game design.

This question utilizes **loops, conditional statements, and switch structures** to demonstrate flow control in solving factorial-based problems.

Task

1. Prompt the user to input a **non-negative integer (n)**.
2. Validate the input, the number must be between **0 and 20**.
3. Calculate the **factorial** using a **loop** (no recursion or built-in factorial functions).
4. Allow the user to choose an application mode:
    - **Combinatorics Mode:** Calculate the number of ways to arrange *n* unique items.
    - **Security Mode:** Estimate unique password combinations using *n* characters.
    - **Game Mode:** Determine how many unique level arrangements can be made from *n* elements.
5. Display the factorial result and an interpretation message based on the selected mode.

Sample Input

```
Enter a number: 5
Choose mode (1-Combinatorics, 2-Security, 3-Game): 2
```

Sample Output

```
Factorial of 5 is 120
In Security Mode: You can create 120 unique password combinations using 5 characters.
Tip of the Day: Factorials grow very fast; optimize your code for efficiency!
```

Constraints

- $0 \leq n \leq 20$
- Input must be an integer.
- Invalid inputs (negative, non-integer, or too large) should display an error message: "Invalid input! Please enter a number between 0 and 20."
- Use **loops** for factorial computation.
- Use a **switch** or **if-else** structure for mode selection.
- Display a short "Tip of the Day" message to encourage learning.

Q5. Space Mission Fuel Efficiency Calculator

As part of an interplanetary space mission simulation, NASA’s junior engineers are tasked to calculate the **fuel efficiency** of different spacecrafts based on their distance traveled and total fuel consumed. The goal is to ensure the best use of fuel resources while keeping astronauts safe and the mission cost-efficient.

This lab question helps you apply **loops, conditionals, and switch structures** to calculate, compare, and evaluate performance based on user inputs.

Task

1. Prompt the user to input:
    - **Spacecraft name**
    - **Distance traveled (in kilometers)**
    - **Fuel used (in liters)**

Calculate **fuel efficiency** using the formula:
efficiency = distance/fuel

2. Validate the input values to ensure:
    - Distance and fuel are **positive numbers**.
    - Fuel is **not zero** (to prevent division errors).
3. Classify the spacecraft’s efficiency using **if-else conditions**:
    - efficiency > 10 → “Outstanding performance! Minimal fuel usage.”
    - 5 ≤ efficiency ≤ 10 → “Good performance! Efficient mission.”
    - efficiency < 5 → “Needs improvement! High fuel consumption.”
4. Add a **switch structure** to display the spacecraft type based on user selection:
    - 1 → Rocket
    - 2 → Shuttle
    - 3 → Probe
5. Use a loop to allow multiple spacecraft entries. Stop when the user types "STOP".
6. Display a short “Mission Tip” message randomly chosen using switch logic.

Sample Input

```
Enter spacecraft name: Apollo-11

Enter distance traveled (km): 250000

Enter fuel used (liters): 20000

Select spacecraft type (1-Rocket, 2-Shuttle, 3-Probe): 1
```

Sample Output

```
Spacecraft: Apollo-11
Type: Rocket
Fuel Efficiency: 12.5 km/l
Performance: Outstanding performance! Minimal fuel usage.
Mission Tip: Always monitor fuel ratios before takeoff!
```

Constraints

- distance > 0
- fuel > 0
- efficiency should be a positive decimal value
- Invalid or zero inputs should display:
  "Invalid input! Distance and fuel must be positive numbers."
- Must use **loops** for repeated entries.
- Must use **if-else** for performance classification.
- Must use **switch** for spacecraft type and "Mission Tip" message selection.

Q8. Fitness Tracker Challenge

Staying active is an important part of maintaining a healthy lifestyle. In this lab, you will create a simple fitness tracker program that records the user's daily steps for one week and provides feedback on their overall activity. The program will help users track their progress, identify their most active day, and stay motivated with encouraging messages.

This question utilizes **loops**, **input validation**, and **conditional statements** to demonstrate how flow control can be used in real-world health and wellness applications.

Task

1. Prompt the user to enter the number of steps taken for each of the 7 days in a week.
2. Validate that each input is a non-negative integer and within a realistic range (0–50,000).
3. Calculate and display:
   - The **total steps** for the week
   - The **average daily steps**
   - The **most active day** (the day with the highest number of steps)
4. Display a motivational message based on the total steps:
   - **≥ 70,000** → "Amazing! You're smashing your goals!"
   - **40,000–69,999** → "Good job! Keep pushing for consistency!"
   - **< 40,000** → "You can do better, small steps daily make a big difference!"
5. Add a random "Health Tip of the Day" using a **switch structure** for creativity.

Sample Input

```
Enter steps for Day 1: 9500

Enter steps for Day 2: 10200

Enter steps for Day 3: 8700

Enter steps for Day 4: 12000

Enter steps for Day 5: 9800

Enter steps for Day 6: 10400

Enter steps for Day 7: 9100
```

Sample Output

```
Total Steps This Week: 69700
Average Daily Steps: 9957.14
Most Active Day: Day 4 with 12000 steps
Good job! Keep pushing for consistency!
Tip of the Day: Stay hydrated and stretch after walks!
```

Constraints

- 0 ≤ steps ≤ 50,000
- Input must be an integer.
- Invalid inputs should display an error message:
  "Invalid input! Please enter a number between 0 and 50,000."
- Use **loops** to handle input and calculations.
- Use a **switch** structure for the random tip selection.